UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/964,950	09/27/2001	Sanjiv M. Shah	42390P11914	2549

8791 7590 03/29/2005

BLAKELY SOKOLOFF TAYLOR & ZAFMAN
12400 WILSHIRE BOULEVARD
SEVENTH FLOOR
LOS ANGELES, CA 90025-1030

EXAMINER	
BULLOCK JR, LEWIS ALEXANDER	
ART UNIT	**PAPER NUMBER**
2195	

DATE MAILED: 03/29/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/964,950	SHAH ET AL.	
	Examiner	Art Unit	
	Lewis A. Bullock, Jr.	2127	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☐ Responsive to communication(s) filed on _____.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) 1-24 is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) 1-24 is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on 27 September 2001 is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date 1/9/02.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

Claim Rejections - 35 USC § 101

1. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

Claims 1-9 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter. M.P.E.P. 2105, section IV states: .

1. Nonstatutory Subject Matter

Claims to computer-related inventions that are clearly nonstatutory fall into the same general categories as nonstatutory claims in other arts, namely natural phenomena such as magnetism, and abstract ideas or laws of nature which constitute "descriptive material." Abstract ideas, *Warmerdam*, 33 F.3d at 1360, 31 USPQ2d at 1759, or the mere manipulation of abstract ideas, *Schrader*, 22 F.3d at 292-93, 30 USPQ2d at 1457-58, are not patentable. Descriptive material can be characterized as either "functional descriptive material" or "nonfunctional descriptive material." In this context, "functional descriptive material" consists of data structures and computer programs which impart functionality when employed as a computer component. (The definition of "data structure" is "a physical or logical relationship among data elements, designed to support specific data manipulation functions." The New IEEE Standard Dictionary of Electrical and Electronics Terms 308 (5th ed. 1993).) "Nonfunctional descriptive material" includes but is not limited to music, literary works and a compilation or mere arrangement of data.

Both types of "descriptive material" are nonstatutory when claimed as descriptive material *per se*. *Warmerdam*, 33 F.3d at 1360, 31 USPQ2d at 1759. When functional descriptive material is recorded on some computer-readable medium it becomes structurally and functionally interrelated to the medium and will be statutory in most cases since use of technology permits the function of the descriptive material to be realized. Compare *In re Lowry*, 32 F.3d 1579, 1583-84, 32 USPQ2d 1031, 1035 (Fed. Cir. 1994) (claim to data structure stored on a computer readable medium that increases computer efficiency held statutory) and *Warmerdam*, 33 F.3d at 1360-61, 31 USPQ2d at 1759 (claim to computer having a specific data structure stored in memory held statutory product-by-process claim) with *Warmerdam*, 33 F.3d at 1361, 31 USPQ2d at 1760 (claim to a data structure *per se* held nonstatutory). When nonfunctional descriptive material is recorded on some computer-readable medium, it is not statutory since no requisite functionality is present to satisfy the practical application requirement. Merely claiming nonfunctional descriptive material stored in a computer-readable medium does not make it statutory. Such a result would exalt form over substance. *In re Sarkar*, 588 F.2d 1330, 1333, 200 USPQ 132, 137 (CCPA 1978) ("[E]ach invention must be evaluated as claimed; yet semantogenic considerations preclude a determination based solely on words appearing in the claims. In the final analysis under 101, the claimed invention, as a whole, must be evaluated for what it is.") (quoted with approval in *Abele*, 684 F.2d at 907, 214 USPQ at 687). See also *In re Johnson*, 589 F.2d 1070, 1077, 200 USPQ 199, 206 (CCPA 1978) ("form of the claim is often an exercise in drafting"). Thus, nonstatutory music is not a computer component and it does not become statutory by merely recording it on a compact disk. Protection for this type of work is provided under the copyright law.

Claims to processes that do nothing more than solve mathematical problems or manipulate abstract ideas or concepts are more complex to analyze and are addressed below.

If the "acts" of a claimed process manipulate only numbers, abstract concepts or ideas, or signals representing any of the foregoing, the acts are not being applied to appropriate subject matter. *Schrader*, 22 F.3d at 294-95, 30 USPQ2d at 1458-59. Thus, a process consisting solely of mathematical operations, i.e., converting one set of numbers into another set of numbers, does not manipulate appropriate subject matter and thus cannot constitute a statutory process.

(a) Functional Descriptive Material: "Data Structures" Representing Descriptive Material *Per Se* or Computer Programs Representing Computer Listings *Per Se*

Data structures not claimed as embodied in computer-readable media are descriptive material *per se* and are not statutory because they are not capable of causing functional change in the computer. See, e.g., *Warmerdam*, 33 F.3d at 1361, 31 USPQ2d at 1760 (claim to a data structure *per se* held nonstatutory). Such claimed data structures do not define any structural and functional interrelationships between the data structure and other claimed aspects of the invention which permit the data structure's functionality to be realized. In contrast, a claimed computer-readable medium encoded with a data structure defines structural and functional interrelationships between the data structure and the computer software and hardware components which permit the data structure's functionality to be realized, and is thus statutory.

Similarly, computer programs claimed as computer listings *per se*, i.e., the descriptions or expressions of the programs, are not physical "things." They are neither computer components nor statutory processes, as they are not "acts" being performed. Such claimed computer programs do not define any structural and functional interrelationships between the computer program and other claimed elements of a computer which permit the computer program's functionality to be realized. In contrast, a claimed computer-readable medium encoded with a computer program is a computer element which defines structural and functional interrelationships between the computer program and the rest of the computer which permit the computer program's functionality to be realized, and is

thus statutory. Accordingly, it is important to distinguish claims that define descriptive material *per se* from claims that define statutory inventions.

Computer programs are often recited as part of a claim. Office personnel should determine whether the computer program is being claimed as part of an otherwise statutory manufacture or machine. In such a case, the claim remains statutory irrespective of the fact that a computer program is included in the claim. The same result occurs when a computer program is used in a computerized process where the computer executes the instructions set forth in the computer program. Only when the claimed invention taken as a whole is directed to a mere program listing, i.e., to only its description or expression, is it descriptive material *per se* and hence nonstatutory.

Since a computer program is merely a set of instructions capable of being executed by a computer, the computer program itself is not a process and Office personnel should treat a claim for a computer program, without the computer-readable medium needed to realize the computer program's functionality, as nonstatutory functional descriptive material. When a computer program is claimed in a process where the computer is executing the computer program's instructions, Office personnel should treat the claim as a process claim. See paragraph IV.B.2(b), below. When a computer program is recited in conjunction with a physical structure, such as a computer memory, Office personnel should treat the claim as a product claim. See paragraph IV.B.2(a), below.

The cited claims 1-9 all detail software instructions that are rejected based upon 35 U.S.C. 101 as nonstatutory functional descriptive material. As detailed above the cited instructions would have to be, "executed by a computer" or "embedded on a computer-readable medium to be executed by a computer to enable the computer to function..", in order to consider statutory process claims. In addition, simply stating the method to be embedded on a computer readable medium **without** the limitations of "when executed by a set of one or more processors, to cause said set of processors to perform operations comprising" is insufficient since the specification details that a computer readable medium is a data signal, e.g. carrier waves, infrared signals, digital signals. Each of the cited signals are not tangible. Therefore, the examiner suggest that

Applicant amend the claims such that the method conforms to the statutory machine-readable medium as described in other pending claims.

Claim Rejections - 35 USC § 112

2. The following is a quotation of the first paragraph of 35 U.S.C. 112:

> The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

3. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

M.P.E.P. 2174 states:

2174 Relationship Between the Requirements of the First and Second Paragraphs of 35 U.S.C. 112

The requirements of the first and second paragraphs of 35 U.S.C. 112 are separate and distinct. If a description or the enabling disclosure of a specification is not commensurate in scope with the subject matter encompassed by a claim, that fact alone does not render the claim imprecise or indefinite or otherwise not in compliance with 35 U.S.C. 112, second paragraph; rather, the claim is based on an insufficient disclosure (35 U.S.C. 112, first paragraph) and should be rejected on that ground. *In re Borkowski*, 422 F.2d 904, 164 USPQ 642 (CCPA 1970). If the specification discloses that a particular feature or element is critical or essential to the practice of the invention, failure to recite or include that particular feature or element in the claims may provide a basis for a rejection based on the ground that those claims are not supported by an enabling disclosure. *In re Mayhew*, 527 F.2d 1229, 188 USPQ 356 (CCPA 1976). In *Mayhew*, the examiner argued that the only mode of operation of the process disclosed in the specification involved the use of a cooling zone at a particular location in the processing cycle. The claims were rejected because they failed to specify either a cooling step or the location of the step in the process. The court was convinced that the cooling bath and its location were essential, and held that claims which failed to recite the use of a cooling zone, specifically located, were not supported by an enabling disclosure (35 U.S.C. 112, first paragraph).

4. Claims 1-7, 9-22 and 24 are rejected under 35 U.S.C. 112, first paragraph, as based on a disclosure which is not enabling. "The determination of if the task was undertaken by another thread by comparing the thread's private values to see if one value is greater than the other" is critical or essential to the practice of the invention, but not included in the claim(s) is not enabled by the disclosure. See *In re Mayhew*, 527 F.2d 1229, 188 USPQ 356 (CCPA 1976). Applicants invention is to correct the inefficiencies of the prior techniques of the SINGLE construct, wherein each thread must go through a lock acquisition phase for each instance of the SINGLE construct (see specification pg. 2, paragraph 0004 – pg. 3, paragraph 0006). This is achieved by each thread maintaining two private values, one of which is a boundary for accessing the global shared object and correlates to the global shared value, the other indicates instance arrived to such that the determination of whether a task was undertaken is based on the comparing of the private values of the thread such that the expensive task (i.e. a task requiring long execution time) does not repeatedly undergo a lock acquisition phase for each encountered thread. Therefore, fewer lock acquisitions result in more efficient execution of the code (specification pg. 10, paragraph 0034). The cited claims detail that the determination and comparison is between the shared value and a private value. This is similar to the prior art techniques of each thread lock the shared value in order to determine if a task was processed. Therefore, Applicant is not claiming the critical element for the invention such that fewer lock acquisitions occur. Applicant on the other hand is claiming the exact opposite of the cited invention, herein the admitted

prior art technique, in the cited claims by not including the critical element as disclosed above.

5. Claims 1-7, 9-22 and 24 are rejected under 35 U.S.C. 112, second paragraph, as being incomplete for omitting essential elements, such omission amounting to a gap between the elements. See MPEP § 2172.01. The omitted elements are: "The determination of if the task was undertaken by another thread by comparing the thread's private values to see if one value is greater than the other" is critical or essential to the practice of the invention, but not included in the claim(s) is not enabled by the disclosure. See *In re Mayhew*, 527 F.2d 1229, 188 USPQ 356 (CCPA 1976). The examiner refers to the statement made in the 35 U.S.C. 112, first paragraph rejection in making the 35 U.S.C. 112, second paragraph rejection.

Claim Rejections - 35 USC § 103

6. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

7. Claims 1-12 and 16-24 are rejected under 35 U.S.C. 103(a) as being unpatentable over "Synchronization Constructs for Parallel Fortran" by IBM in view of Applicant's Admitted Prior Art (APA).

As to claim 1, IBM teaches a method comprising: receiving a first code segment (parallel construct / parallel loop / extended constructs), the first code segment having a set of instances (iterations) of a parallel construct (pg. 3, "The parallel loop allows concurrent processes to perform the task of the look in parallel, by assigning successive values of the loop index to various processes and letting each process doe one iteration at a time."; pg. 3, "The initial implementation is in FORTRAN, for which a preprocessor translates the extended constructs into FORTRAN code."), each of the set of instances (iterations) of the parallel construct (parallel construct / parallel loop) comprised of a task (task) (pg. 3, "The parallel loop allows concurrent processes to perform the task of the look in parallel, by assigning successive values of the loop index to various processes and letting each process doe one iteration at a time."); and translating the first code segment (initial implementation in FORTRAN) to a second code segment (extended constructs translated into FORTRAN code) (pg. 3, "The initial implementation is in FORTRAN, for which a preprocessor translates the extended constructs into FORTRAN code."), the second code segment, when being executed to perform operations comprising: allocating a shared value (shared index), the shared value to indicate a most current one of the set of instances (iterations) encountered by one of a team of processes (processes) (pg. 5, "The availability of work is inferred from the value of the shared index"), allocating a private value (private index) for each of the team of processes (processes), the private value to indicate one of the set of instances encountered by the private value's corresponding process of the team of processes (processes) (pg. 5, "When a process arrives at the beginning of the loop, its private

loop-index is assigned a value, and the process performs an iteration of the loop."), maintaining the shared value with the team of processes (via the processes access the shared index to ascertain more work) (pg. 5, "At the end of the loop, the process returns to the beginning of the loop to be assigned another value of the loop index...The assignment of index values is managed by a fetch-and-add location, called the shared index....The availability of work is inferred from the value of the shared index. When all values of the shared index have been assigned, there is no more work to be assigned."), and maintaining the private value (private index) of each of the team of processes (processes) with the private value's corresponding process of the team of processes (pg. 4, "The loop index must be a private integer variable, not a shared variable."). However, IBM does not teach explicitly state that the processes are threads.

APA teaches the OpenMP specification provides various work sharing constructs for parallelization of programs with a team of threads (pg. 2, paragraph 0003 – 0004). It is obvious to one skilled in the art at the time of the invention that the team of threads of APA would function as the team of processes of IBM in order to process and execute the task of a parallel construct and therefore, would be obvious to combine the teachings of IBM with the teachings of APA to allow parallel execution in situations where threads execute a parallel construct which there is only a small amount of work involved in any one loop iteration, and the overhead involved in accessing the shared index repeatedly for each iteration would dominate the execution time (pg. 4 of IBM reference).

As to claim 6, IBM teaches a method comprising: receiving a first code segment (parallel construct / parallel loop / extended constructs), the first code segment having a set of instances (iterations) of a parallel construct (pg. 3, "The parallel loop allows concurrent processes to perform the task of the look in parallel, by assigning successive values of the loop index to various processes and letting each process doe one iteration at a time."; pg. 3, "The initial implementation is in FORTRAN, for which a preprocessor translates the extended constructs into FORTRAN code."), each of the set of instances (iterations) being associated with a task (task) (pg. 3, "The parallel loop allows concurrent processes to perform the task of the look in parallel, by assigning successive values of the loop index to various processes and letting each process doe one iteration at a time."); and for each of the set of instances, generating a second code segment (via preprocessor) (pg. 3, "The initial implementation is in FORTRAN, for which a preprocessor translates the extended constructs into FORTRAN code."), which when executed, cause a first process to perform operations comprising: encountering one of the set of instances (pg. 5, "When a process arrives at the beginning of the loop, its private loop-index is assigned a value, and the process performs an iteration of the loop."; "...the process returns to the beginning of the loop to be assigned another value of the loop index."), determining if the task of the one of the set of instances (iterations) has been undertaken by a second process (via determining whether all values of the shared index have been assigned), upon determining the task has not been undertaken by the second process (via all values of the shared index have not been assigned), undertaking the task (task) and indicating with a shared value (via incrementing the

shared value) the undertaking of the task (task), indicating with a first private value (private value) of the first process the one of the set of instances (iteration) (pg. 5, "The assignment of index values is managed by a fetch-and-add location, called the shared index) (pg. 5, "At the end of the loop, the process returns to the beginning of the loop to be assigned another value of the loop index...The assignment of index values is managed by a fetch-and-add location, called the shared index....The availability of work is inferred from the value of the shared index. When all values of the shared index have been assigned, there is no more work to be assigned."); and maintaining a second private value (private clock) of the first process, the second private value being a boundary for accessing the shared value (shared index / loop index) (pg. 7, "An additional requirement that we address here, but which is not specifically addressed in other implementations, is that processes that are delayed and arrive at a loop after it has been reset must bypass the loop and continue on, rather than commence work on a new execution of the loop...On arriving at a parallel construct, each process compares its private clock with the public clock. If it arrived late, the construct is bypassed. If it arrived early, it must wait until the loop is reset for the desired execution of the loop."). However, IBM does not teach explicitly state that the processes are threads.

APA teaches the OpenMP specification provides various work sharing constructs for parallelization of programs with a team of threads (pg. 2, paragraph 0003 – 0004). It is obvious to one skilled in the art at the time of the invention that the team of threads of APA would function as the team of processes of IBM in order to process and execute the task of a parallel construct and therefore, would be obvious to combine the

teachings of IBM with the teachings of APA to allow parallel execution in situations where threads execute a parallel construct which there is only a small amount of work involved in any one loop iteration, and the overhead involved in accessing the shared index repeatedly for each iteration would dominate the execution time (pg. 4 of IBM reference).

As to claim 10, IBM teaches an apparatus comprising: a memory to host a code segment (parallel construct / parallel loop / extended constructs), the code segment having a set of instances (iteratons) of a parallel construct (pg. 3, "The parallel loop allows concurrent processes to perform the task of the look in parallel, by assigning successive values of the loop index to various processes and letting each process doe one iteration at a time."; pg. 3, "The initial implementation is in FORTRAN, for which a preprocessor translates the extended constructs into FORTRAN code."), each of the set of instances of the parallel construct associated with a task (task) (pg. 3, "The parallel loop allows concurrent processes to perform the task of the look in parallel, by assigning successive values of the loop index to various processes and letting each process doe one iteration at a time."); and a set of processors (processors) (pg. 2, "If many processes are assigned to a program and it is run on a multiprocessor system...It is based on a shared memory organization.") coupled to the memory, the set of processors (processors) to process the code segment (pg. 2, "If many processes are assigned to a program and it is run on a multiprocessor system...It is based on a shared memory organization."), to encounter one of the set of instances (iterations) while

processing the code segment, to generate a second code segment corresponding to the one of the set of instances (via preprocessor) (pg. 3, "The initial implementation is in FORTRAN, for which a preprocessor translates the extended constructs into FORTRAN code."), the second code segment (FORTRAN code) to cause each of the team of processes to perform the following: to determine if the task associated with the one of the set of instances (iterations) has been undertaken by another one of the set of processes (via determining whether all values of the shared index have been assigned), if the task has not been undertaken by another one of the set of processes, to undertake the task and to indicate with a shared value (shared index) the undertaking of the task (via incrementing the shared index), and to indicate with a private value (private index) the one of the set of instances (via a fetch-and-add location) (pg. 5, "The assignment of index values is managed by a fetch-and-add location, called the shared index) (pg. 5, "At the end of the loop, the process returns to the beginning of the loop to be assigned another value of the loop index...The assignment of index values is managed by a fetch-and-add location, called the shared index....The availability of work is inferred from the value of the shared index. When all values of the shared index have been assigned, there is no more work to be assigned."; pg. 4, "The loop index must be a private integer variable, not a shared variable."). However, IBM does not teach explicitly state that the processes are threads.

APA teaches the OpenMP specification provides various work sharing constructs for parallelization of programs with a team of threads (pg. 2, paragraph 0003 – 0004). It is obvious to one skilled in the art at the time of the invention that the team of threads of

APA would function as the team of processes of IBM in order to process and execute the task of a parallel construct and therefore, would be obvious to combine the teachings of IBM with the teachings of APA to allow parallel execution in situations where threads execute a parallel construct which there is only a small amount of work involved in any one loop iteration, and the overhead involved in accessing the shared index repeatedly for each iteration would dominate the execution time (pg. 4 of IBM reference).

As to claim 2, IBM teaches wherein the process undertakes the task (task) of one of the set of instances (iterations) if the shared value (shared index) and the private value (private index) indicate a same one of the set of instances (via being assigned another value of the loop index from the shared index, thereby more work is assigned) (pg. 5).

As to claim 3, IBM teaches wherein maintaining the shared value (shared index) comprises incrementing the shared value (shared index) when one of the team of processes undertakes the task of one of the set of instances (iterations) (via being assigned another value of the loop index) (pg. 5).

As to claim 4, IBM teaches maintaining the private value (private index) comprises incrementing the private value (private index) when the private's value's corresponding process of the team of processes encounters one of the set of instances

(iterations) (via the private loop-index is assigned a value; and at the end of the loop, the process returns to the beginning of the loop to be assigned another value of the loop index wherein the assignments are performed via a fetch-and-add location called the shared index wherein the shared index value is incremented via each fetch instruction) (pg. 5).

As to claim 5, IBM teaches the process maintaining a second private value (private clock), the second private value being a boundary for accessing the shared value (shared index / loop index) (pg. 7, "An additional requirement that we address here, but which is not specifically addressed in other implementations, is that processes that are delayed and arrive at a loop after it has been reset must bypass the loop and continue on, rather than commence work on a new execution of the loop...On arriving at a parallel construct, each process compares its private clock with the public clock. If it arrived late, the construct is bypassed. If it arrived early, it must wait until the loop is reset for the desired execution of the loop.").

As to claim 7, IBM teaches determining if the task (task) of the one of the set of instances (iterations) has been undertaken by the second process comprises the first process comparing the shared value (shared index) and its first private value (private index) (via the process retrieving a shared index value and store the value in its private value to be assigned more work) (pg. 5, "At the end of the loop, the process returns to the beginning of the loop to be assigned another value of the loop index...The

assignment of index values is managed by a fetch-and-add location, called the shared index....The availability of work is inferred from the value of the shared index. When all values of the shared index have been assigned, there is no more work to be assigned.").

As to claim 8, IBM teaches determining if the task (task) of the one of the set of instances (iterations) has been undertaken by the second process (process) comprises the first process comparing the first private value (private index) of the first process and the second private value (private clock) of the first process (via the process trying to acquire a new index via determining whether the shared index has more values and if the iterations can be processed based on whether the process arrived early or late to the parallel construct) (pg. 5 and pg. 7).

As to claim 9, IBM teaches the second process maintains a third private value (private index) and a fourth private value (clock value), the third private value indicating a second one of the set of instances (iteration) encountered by the second process (pg. 5) (pg. 5, "At the end of the loop, the process returns to the beginning of the loop to be assigned another value of the loop index...The assignment of index values is managed by a fetch-and-add location, called the shared index....The availability of work is inferred from the value of the shared index. When all values of the shared index have been assigned, there is no more work to be assigned."); and the fourth private value (clock value) indicating the second process's boundary for accessing the shared value (via the

process trying to acquire a new index via determining whether the shared index has more values and if the iterations can be processed based on whether the process arrived early or late to the parallel construct) (pg. 7, "An additional requirement that we address here, but which is not specifically addressed in other implementations, is that processes that are delayed and arrive at a loop after it has been reset must bypass the loop and continue on, rather than commence work on a new execution of the loop...On arriving at a parallel construct, each process compares its private clock with the public clock. If it arrived late, the construct is bypassed. If it arrived early, it must wait until the loop is reset for the desired execution of the loop.").

As to claim 11, IBM teaches the set of processors (processors) to determine if the task has been undertaken by another one of the set of processes comprises to compare the private value (private index) and the shared value (shared index) (via the process retrieving a shared index value and store the value in its private value to be assigned more work) (pg. 5, "At the end of the loop, the process returns to the beginning of the loop to be assigned another value of the loop index...The assignment of index values is managed by a fetch-and-add location, called the shared index....The availability of work is inferred from the value of the shared index. When all values of the shared index have been assigned, there is no more work to be assigned.").

As to claim 12, IBM teaches the set of processors (processors which execute the processes) to maintain a second private value (private clock), the second private value

being a boundary to access the shared value (via the process trying to acquire a new index via determining whether the shared index has more values and if the iterations can be processed based on whether the process arrived early or late to the parallel construct) (pg. 7, "An additional requirement that we address here, but which is not specifically addressed in other implementations, is that processes that are delayed and arrive at a loop after it has been reset must bypass the loop and continue on, rather than commence work on a new execution of the loop...On arriving at a parallel construct, each process compares its private clock with the public clock. If it arrived late, the construct is bypassed. If it arrived early, it must wait until the loop is reset for the desired execution of the loop.").

As to claims 16-24, reference is made to a machine readable medium that corresponds to the method of claims 1-9 and is therefore met by the rejection of claims 1-9 above.

8. Claims 13-15 are rejected under 35 U.S.C. 103(a) as being unpatentable over "Synchronization Constructs for Parallel Fortran" by IBM in view of Applicant's Admitted Prior Art and "Compilers, Principles, Techniques, and Tools by AHO et al.

As to claim 13, IBM teaches a system comprising: a translating unit (preprocessor) to translate a code, the code having an instance (iteration) of a parallel construct (parallel construct / parallel loop / extended constructs) (pg. 3, "The parallel loop allows concurrent processes to perform the task of the look in parallel, by assigning

successive values of the loop index to various processes and letting each process doe one iteration at a time."; pg. 3, "The initial implementation is in FORTRAN, for which a preprocessor translates the extended constructs into FORTRAN code."), the instance associated with a task (task) (pg. 3, "The parallel loop allows concurrent processes to perform the task of the look in parallel, by assigning successive values of the loop index to various processes and letting each process doe one iteration at a time."), the translating unit to generate a code segment (FORTRAN code), the code segment to cause each of a set of processes to perform the following: to determine if the task associated with the instances (iteration) has been undertaken by another one of the set of (processes) (via determining whether all values of the shared index have been assigned), if the task has not been undertaken by another one of the set of processes (via all values of the shared index have not been assigned), to undertake the task (task) and to indicate with a shared value the undertaking of the task (via incrementing the shared value) the undertaking of the task (task), indicating with a private value (private value) of the first process the instance (iteration) (pg. 5, "The assignment of index values is managed by a fetch-and-add location, called the shared index) (pg. 5, "At the end of the loop, the process returns to the beginning of the loop to be assigned another value of the loop index...The assignment of index values is managed by a fetch-and-add location, called the shared index....The availability of work is inferred from the value of the shared index. When all values of the shared index have been assigned, there is no more work to be assigned."); and a set of processors (processors) coupled to the translating unit (preprocessor), the set of processors to host the set of processes (pg. 2,

"If many processes are assigned to a program and it is run on a multiprocessor system...It is based on a shared memory organization."). However, IBM does not teach explicitly state that the processes are threads and a linker unit coupled to the translating unit.

APA teaches the OpenMP specification provides various work sharing constructs for parallelization of programs with a team of threads (pg. 2, paragraph 0003 – 0004). It is obvious to one skilled in the art at the time of the invention that the team of threads of APA would function as the team of processes of IBM in order to process and execute the task of a parallel construct and therefore, would be obvious to combine the teachings of IBM with the teachings of APA to allow parallel execution in situations where threads execute a parallel construct which there is only a small amount of work involved in any one loop iteration, and the overhead involved in accessing the shared index repeatedly for each iteration would dominate the execution time (pg. 4 of IBM reference).

AHO teaches a preprocessor – compiler that sends code to a linker unit coupled to the translating unit, the linker unit (loader / link-editor) to link the translated code with a library (library) (pg. 4, "The compiler in Fig. 1.3 creates assembly code that is translated by an assembler into machine code and then linked together with some library routines into the code that runs on the machine."). Therefore, it would be obvious to one skilled in the art to combine the teachings of IBM with the teachings of APA and AHO in order to generate code to run on a machine.

As to claim 14, IBM teaches the set of processors (processors) to determine if the task has been undertaken by another one of the set of processes comprises to compare the private value (private index) and the shared value (shared index) (via the process retrieving a shared index value and store the value in its private value to be assigned more work) (pg. 5, "At the end of the loop, the process returns to the beginning of the loop to be assigned another value of the loop index...The assignment of index values is managed by a fetch-and-add location, called the shared index....The availability of work is inferred from the value of the shared index. When all values of the shared index have been assigned, there is no more work to be assigned.").

As to claim 15, IBM teaches the set of processors (processors which execute the processes) to maintain a second private value (private clock), the second private value being a boundary to access the shared value (via the process trying to acquire a new index via determining whether the shared index has more values and if the iterations can be processed based on whether the process arrived early or late to the parallel construct) (pg. 7, "An additional requirement that we address here, but which is not specifically addressed in other implementations, is that processes that are delayed and arrive at a loop after it has been reset must bypass the loop and continue on, rather than commence work on a new execution of the loop...On arriving at a parallel construct, each process compares its private clock with the public clock. If it arrived late, the construct is bypassed. If it arrived early, it must wait until the loop is reset for the desired execution of the loop.").

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Lewis A. Bullock, Jr. whose telephone number is (571) 272-3759. The examiner can normally be reached on Monday-Friday, 8:30 am - 5:00 pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Meng An can be reached on (571) 272-3756. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

LEWIS A. BULLOCK, JR.
PRIMARY EXAMINER

March 14, 2005